# Rethinking Software Network Data Planes in the Era of Microservices

Ph.D. Thesis Summary

**Candidate:** Sebastiano Miano
**Advisor:** Fulvio Risso

With the advent of Software Defined Networks (SDN) and Network Functions Virtualization (NFV), software started playing a crucial role in the computer network architectures, with the end-hosts representing natural enforcement points for core network functionalities that go beyond simple switching and routing. Recently, there has been a definite shift in the paradigms used to develop and deploy server applications in favor of microservices, which has also brought a visible change in the type and requirements of network functionalities deployed across the data center. Network applications should be able to continuously adapt to the runtime behavior of cloud-native applications, which might regularly change or be scheduled by an orchestrator, or easily interact with existing "native" applications by leveraging kernel functionalities - all of this without sacrificing performance or flexibility.

In this dissertation, we start by analyzing the main design problems and the most important characteristics that these new types of applications should have, taking into account the transformation of workloads and the unique requirements needed. Most of the existing networking technologies and frameworks today primarily focus on high-throughput, low-latency, CPU-hungry, and unalterably deployed services, which do not fit with those applications' needs. The current patterns of writing monolithic Virtual Network Functions (VNFs) results contradictory to the cloud-native paradigm, and simply transforming those monolithic patterns into analogous cloud-native counterparts results in manageability, scalability, and performance problems. As a result, most of the current cloud-native solutions rely on fixed functionalities and tools embedded into the operating system network stack, which are subjected to *slow-performance* and *low-innovation*, given the difficulties in maintaining, up-streaming or modifying the kernel code (or the respective kernel modules).

We explore the opportunity to use the extended Berkeley Packet Filter (eBPF) subsystem as a base framework to solve most of the problems mentioned above, and we investigate the possibility to create complex network services that go beyond simple proof-of-concept data plane applications. We then present the most promising characteristics of this technology, as well as the main encountered limitations. The above findings made us reaching to the conclusion that creating network functions based entirely on eBPF is sometimes complicated given the lack of a common framework that

provides useful abstractions to developers to solve common problems and know limitations. Motivated by those findings, we propose a novel network function model where in-kernel network applications can be adjusted, injected, and modified in a simple and defined way. We present Polycube, a framework that brings the power and innovation of NFV services to the world of *in-kernel* packet processing. Polycube provides a standard interface, a set of abstractions, and well-defined APIs to NF developers for both running and developing NFs; its services are dynamically adaptable and can be composed to form in-kernel service chains. We demonstrate the applicability and efficiency of this new model showing the design and evaluation of some real-world applications such as a network plugin for Kubernetes, which provides security and connectivity between cloud-native services, and *bpf-iptables*, a (partial) clone of iptables (one of the most used software nowadays) characterized by an improved classification pipeline, higher speed, and scalability, while preserving the original iptables filtering semantic and compatibility with existing applications; all of this within a vanilla Linux kernel.

Afterward, motivated by the infrastructure efficiency challenges that cloud-scale and cloud-native architectures face today, we investigate the potential of achieving an hardware accelerated and software defined architecture by combining the use of smart network interface cards (SmartNICs) with the eBPF/XDP-based software kernel processing We then analyze and evaluate different conjunctions of hardware and software processing and we propose an architecture, for the DDoS Mitigation use case, that combines the high-speed filtering capabilities of the hardware with the flexibility of packet processing in software to analyze custom properties of the traffic and packets.

Finally, we propose Kecleon, a dynamic compiler framework for building software network data planes that exploits the runtime knowledge of configuration data, data structures content and execution profiles to automatically generate an optimized version of the original data plane that is more efficient for its runtime behavior. Kecleon achieves this result automatically without requiring any user intervention, hence simplifying both the development and manageability of network services, and making it possible to perform some optimizations that are not applicable within a static compilation process. Kecleon optimization pipeline is split into three different phases: an *analysis* phase, used to identify the packet processing logic of the data plane application, an *instrumentation* phase, which is used to gather runtime data about the execution profiles of the NF, and an *optimization* phase, where all the previous information are exploited to apply transformations to both code or data structures that better fits within the runtime environment and behavior of the NF.

In conclusion, we hope that the combination of the works presented in this dissertation can lay the foundation for a new model of packet processing applications that can be dynamically re-combined, re-generated, and re-optimized without sacrificing programmability, extensibility, and performance. We release the source code of all the systems presented in this thesis, and we hope they will be useful for both researchers and industrial system developers to build upon them.